

Symmetry Reduction Enables Model Checking of More Complex Emergent Behaviours of Swarm Navigation Algorithms

Laura Antuña*, Dejanira Araiza-Illan†, Sérgio Campos‡, Kerstin Eder§

June 17, 2015

Abstract

The emergent global behaviours of robotic swarms are important for them to achieve their navigation task goals. These emergent behaviours can be verified to assess their correctness, through techniques like model checking. Model checking exhaustively explores all possible behaviours, based on a discrete model of the system, such as a swarm in a grid. A common problem in model checking is the state-space explosion that arises when the states of the model are numerous. We propose a novel implementation of symmetry reduction, in the form of encoding navigation algorithms relatively with respect to a reference, exploiting the symmetrical properties of swarms in grids. We applied the relative encoding to a swarm navigation algorithm, *Alpha*, modelled for the NuSMV model checker. A comparison of the state-space and verification results with an absolute (or global) and a relative encoding of the *Alpha* algorithm highlights the advantages of our approach, allowing model checking both larger grid sizes and higher numbers of robots, and consequently verifying more complex emergent behaviours. For example, a property was verified for a grid with 3 robots and a maximum allowed size of 8×8 cells in a global encoding, whereas this size was increased to 16×16 using a relative encoding. Also, the time to verify a property for a swarm of 3 robots in a 6×6 grid was reduced from almost 10 hours to only 7 minutes. Our approach is transferable to other swarm navigation algorithms.

*Computer Science Department, Universidade Federal de Minas Gerais, Brazil
laura.antuna@dcc.ufmg.br

†Computer Science Department, University of Bristol, UK
dejanira.araizaillan@bristol.ac.uk

‡Computer Science Department, Universidade Federal de Minas Gerais, Brazil
scampos@dcc.ufmg.br

§Computer Science Department, University of Bristol, UK
kerstin.eder@bristol.ac.uk

1 Introduction

Robotic swarms consist of a set of robots with simple individual behaviour rules, working together in cooperation to achieve a more complex or emergent final behaviour. Appealing characteristics of swarms are the low cost incurred in producing the robots, which have a simple hardware design, scalability, and fault tolerance [7]. Examples of their application to real-life tasks include nanorobotics, disaster rescue missions, and mining or agricultural foraging tasks.

The emergent behaviours of a swarm of robots need to be *verified*, with respect to safety and liveness requirements [19, 20], and *validated* to determine whether it is fit for purpose in the target environment. Safety requirements are the allowed behaviours of the system, and liveness requirements specify the dynamic behaviours expected to happen during the execution of the system [20]. Verification methods include testing over the real robotic platforms or in simulation, and formal, such as model checking and theorem proving. In model checking all the possible behaviours of a system are exhaustively explored to determine whether the requirements are satisfied or violated. Model checking has previously been employed to verify robotic swarms [6, 7, 8, 13, 15].

For model checking the system is modelled in a finite-state manner. The continuous space in which the robots in the swarm move represents a challenge, since it can cause an infinite state model, which translates into a state-space explosion problem when this model is used for verification; i.e., the number of states to explore is beyond computational capabilities. The discretization of the continuous space into cells of fixed size —i.e., a grid— is a solution that has been applied in swarms, to enable model checking [8, 14]. Even with the discretization of the environment into a “small” grid (e.g., 4×4 cells), the state-space explosion problem can occur due to the presence of other variables, which results in too many possible configurations of the robots in the grid.

Symmetry reduction techniques have been used to reduce the size of the models in model checking [1, 2, 5, 9, 10, 11, 12, 18]. These techniques compute a subset of representatives of all the states, after the user provides the classification criteria for the grouping. Alternatively, the classification is computed by analysing similarities amongst the states. Our proposed solution to the state-space explosion problem for swarms in a grid is to exploit the symmetry of the configurations of the robots in the grid, implementing symmetry reduction in a novel manner.

In this paper, we explore the vertical and horizontal symmetry in the grid to reduce the size of the finite-state model. We implemented a *relative encoding* of a swarm environment model that eliminates symmetrically equivalent states from the state space. The swarm is assumed to be homogeneous; i.e., all the robots are considered identical in capabilities and rank. In the relative encoding, one robot is set as the “reference”, with a fixed location and direction of motion. The other robots’ locations and directions are defined based on this reference. In a global or absolute encoding, if all the robots in the grid are simultaneously rotated in the same direction and shifted horizontally or vertically by the same distance, the robots’ new configurations change in location and direction. In

a relative encoding, the locations and directions would remain the same since they are encoded relative to the reference robot, resulting in a reduction of the state space of direction and position configurations.

We applied our approach to the *Alpha* swarm algorithm [17], modelled in the NuSMV model checker [3]. An abstraction of the *Alpha* algorithm is proposed in [7, 8] and verified through model checking against Linear Temporal Logic (LTL) properties. However, further analysis for grids of more than 8×8 cells and 3 robots [7, 8] could not be performed due to state-space explosion. Verification results for larger grids and number of robots had to be extrapolated. In [16], a different abstraction for the *Alpha* algorithm is proposed, using parametrized interleaved interpreted systems, a semantics developed for the verification of multi-agent systems. Expressive temporal-epistemic specifications are verified by parametrizing the model on the number of robots. A “cut-off” that represents the behaviour of swarms of any size can be identified. Although our solution does not scale as well as [16] in terms of the number of robots, it scales very well with respect to the grid size, and thus complements [16] in that aspect.

Firstly, we modified the models in [7, 8] to be relative. The state reduction allowed us to check the same swarm property for a larger number of robots and grid size. Secondly, we abstracted the *Alpha* algorithm in our own terms, employing an explicit collision avoidance mechanism, and modelled it using the relative encoding approach. This new abstraction, despite having more variables, was of a reduced order of states compared to the global encoding in [7, 8]. The reduction allowed us to check the swarm for the same swarm configurations used in [7, 8], but obtaining different verification results. The encoding of this new abstraction shows the potential of applying the relative concept to other swarm navigation algorithms in the same manner, which we will be exploring in the future.

The structure of this paper is as follows. Section 2 introduces grid-based discrete models for swarm robotics. Section 3 presents an overview of model checking, the state-space explosion problem and related symmetry reduction techniques. Section 4 presents the relative encoding model. Section 5 introduces the *Alpha* algorithm. In Section 6 we present, compare and discuss the results of the three encodings of the *Alpha* algorithm, using the abstraction in [7, 8] and ours. The concluding remarks are presented in Section 7.

2 Swarms in Grids

When modelling navigating algorithms for swarms, a critical aspect is the continuous space in which the robots in the swarm act. A common approach is to discretize this environment into squared cells of the same size, forming a grid. This grid can “wrap around”, i.e., it works as if it was projected over a sphere.

Another aspect in the modelling of a swarm is the concurrency of its elements. Four main types have been proposed [8]: *synchrony*, where all robots move at the same time in each step; *strict turn taking*, where only one robot moves at a time, following a strict order; *non-strict turn taking*, where only one

robot moves at a time in a random order, but all the robots get the chance to move after a number of steps; and *fair asynchrony*, where robots move at different time in a random order, and the only guarantee is that a robot will always eventually move.

3 Model Checking and the State-space Explosion Problem

Model checking is a formal verification method. An exhaustive traversal of the reachable states of a discrete model of the system is performed to check the validity of some desired properties, such as liveness and safety. The reachability depends on the allowed transitions from state to state. The properties to be verified are defined using a temporal logic, for example Linear Temporal Logic (LTL). Model checking is fully automatic. Counterexamples can be produced in the case of a property being false, which helps discovering the reason of the failure [4].

Model checkers can be either explicit or symbolic, the latter having an internal structure such as a Binary Decision Diagram (BDD) or Boolean functions that implicitly represent the transitions within the states of the system. The BDDs are constructed before the traversal of the model. Explicit-state model checkers traverse the model whilst verifying it, which may lead to running out of memory before finishing the traversal. Symbolic model checkers allow an initial compression of the model, at the cost of an overhead and memory usage before the checking. NuSMV is an open-source symbolic model checker, with its own input language [3].

The number of states and transitions to traverse in the model can cause problems for model checking (called state-space explosion). Different techniques have been incorporated to alleviate this issue, for instance the use of BDDs that led to the branch of symbolic model checking, symmetry reduction, and abstractions of the model to reduce the number of states [4].

The symmetrical properties of a finite-state model [4] can be identified and the state space of the model reduced before model checking. For example, a “static channel diagram” of a Promela model is computed in [9]. In [12], symmetrical components are reduced by hand, by annotating the model with the directive **TRANS** to eliminate equivalent transitions from state to state. This manual approach is not trivially transferable to reducing the model of a swarm in a grid.

Symmetry reduction can be applied to BDDs [4]. “Quotient models” are proposed in [5, 11]. Automorphisms that preserve the same transition relations in a BDD (or “orbit relations”) are computed from permutations of the states, and a chosen representative state substitutes all the states in each orbit relation, forming a reduced quotient model instead of the original BDD.

Symmetry reduction techniques have also been applied to explicit-state model checkers. In [18], a new data type, “scalarsets”, is added to the input language

of a model checker, to create automorphisms and a “quotient graph” (representatives of groups of states) whilst traversing the model. Scalarsets are similar to the static channel diagram model in [9]. The automorphisms can be computed on-the-fly along with the traversal of the model, as in [2] based on heuristics. The main disadvantage of all these explicit-state and BDD based symmetry reduction methods is that they are applied into the algorithms of the model checker software or BDD computation, which becomes a non-trivial software re-implementation task.

Our approach to avoid the state-space explosion problem in model checking is based on exploiting symmetrical properties of a swarm in a grid. The encoding of the model in a relative manner with respect to a reference point, as opposed to a global or absolute encoding, can be interpreted as the combination of symmetry reduction and abstraction. The proposed approach is analogous to finding orbit relations or representatives in the global model. The relative encoding employs representatives of the global encoding as possible states, i.e., configurations of the robots in the grid.

4 Relative Encoding of a Swarm in a Grid

In swarms, the focal point is the interaction of the elements through time. With only the swarm’s overall behaviour in mind, the swarm moving north or south, with the same distance and orientation between all the robots and environment elements, such as bounds and obstacles, represents essentially the same situation. Furthermore, these two behaviours correspond to the same swarm configuration if the swarm is encoded in a relative manner. A simple relative encoding is to set a robot as the reference, with fixed location and direction, and to base the location of other robots and environment elements on this reference. A grid populated with robots, rotated and shifted horizontally and vertically, results in different values for the direction and position of each robot in a global or absolute encoding. However, in a relative encoding some rotations and shifting motions correspond to the same grid configurations.

If a model with r robots in a $m \times m$ size grid (locations), with d possible directions, p other robots’ variables of domain sizes $v_i, i = 1, \dots, p$, and q global variables of domain sizes $s_j, j = 1, \dots, q$, is globally encoded, the size of the state space to be explored is $(d \times m^2 \times v_1 \times v_2 \times \dots \times v_p)^r \times (s_1 \times s_2 \times \dots \times s_q)$. In a relative encoding, the reference robot will have fixed location and direction, and the resulting state space will be of size $(v_1 \times v_2 \times \dots \times v_p) \times (d \times m^2 \times v_1 \times v_2 \times \dots \times v_p)^{r-1} \times (s_1 \times s_2 \times \dots \times s_q)$. This corresponds to a reduction of the state space of $d \times m^2$. In practice this bound changes according to the variables used in the relative encoding of an algorithm.

This decrease in the state space would improve the performance of model checking in terms of time and memory usage, when verifying emergent behaviours of the swarm. Moreover, a counterexample in the relative model is equivalent to a class of counterexamples in the global model.

The update of the direction and location of the robots based on the reference

robot’s location motion must be performed in two situations: when the reference robot makes a move to an adjacent cell, and when it combines the motion with a change in its direction (rotation). In the first case, the equivalent of the reference robot moving in a direction is the other robots moving in the opposite direction. For example, if the reference moves north, the other robots move south instead. By following this update rule, the distance and direction relation between the swarm of robots remains the same. When the reference robot moves to another cell and also rotates, the other robots make a turn to an opposing direction and their locations need to be updated, as summarized in Table 1.

Table 1: Location and orientation update after the reference robot changed direction

Reference’s change	Direction change	Location change
$n \rightarrow e$	$n \rightarrow w, s \rightarrow e, e \rightarrow n, w \rightarrow s$	$x' = m - y, y' = x$
$n \rightarrow s$	$n \rightarrow s, s \rightarrow n, e \rightarrow w, w \rightarrow e$	$x' = m - x, y' = m - y$
$n \rightarrow w$	$n \rightarrow e, s \rightarrow w, e \rightarrow s, w \rightarrow n$	$x' = y, y' = m - x$

The relative encoding in NuSMV input language has been designed to have the following structure, to facilitate modularity, employing **MODULE** constructions: (a) we used distances between the reference robot and other robots, instead of specific locations; (b) we defined and encoded the update of the distance variables in the **main** module, along with the concurrency mode logic (e.g., a **turn** variable); and (c) we defined and encoded the motion algorithm (next motion, step size, rotations) of each robot in **robot** modules, which update the values of the variables in the **main** module. This procedure can be partially automated through a script to create the **main** module, from selecting the number of robots and concurrency mode, which we implemented to generate the NuSMV code for the experiments in Section 6. The **robot** modules are encoded manually, as they depend on the navigation algorithm.

5 Alpha Algorithm

The *Alpha* algorithm has been used as a case study to demonstrate how to verify emergent behaviours in swarms through model checking tools, as in [6, 7, 8, 16]. In the *Alpha* algorithm, the robots in the swarm navigate the environment trying to maintain connectivity, defined as a wireless range. This is achieved by the following rules: (a) the default movement of a robot is forward, maintaining its current direction. (b) When a robot loses connection with another robot, if the remaining number of connected robots is smaller than a value α , the robot makes a 180° turn. (c) Every time a robot regains connectivity with another, it performs a random turn.

A critical requirement for a swarm is: *All robots shall eventually be connected*. Expressed formally in LTL, this property was proved to be false in [7, 8].

6 Results

We compared a global encoding of the *Alpha* algorithm based on the abstraction proposed in [7, 8], against a relative encoding of it, both implemented in the input language of NuSMV. Also, we implemented our own abstraction of the *Alpha* algorithm from the ground up, which employs more variables than in [7, 8] for the concurrency modes (strict turn taking, non-strict turn taking, and fair asynchrony), encoded in a relative manner. We measured the state-space reduction of the relative models, compared to the global model.

Table 2 shows the reduction in the reachable states for the *Alpha* algorithm, with grid size 8×8 , 3 robots, and $\alpha = 1$. The state space was computed automatically when compiling the models in NuSMV. Our new abstraction of the *Alpha* algorithm also has fewer states than the global one from [7, 8]. We decided not to consider the fully synchronous concurrency mode in the results, since, in the context of the *Alpha* algorithm, it allows for behaviours that are incorrect in the real world: the robots would be allowed to “swap” their cell locations.

For the strict turn taking concurrency mode, for example, all three encodings contain a variable `turn` declared in the `main` module (a global variable), of domain size 3, and variables `x`, `y` and `direction` declared in the `robot` module (robot variables), of domain sizes 8, 8 and 4, respectively. Both the global abstraction from [7, 8] and its relative encoding contain a `motion` variable with domain size 2. The relative encoding also contains a global variable `random`. Our new abstraction includes the global variables `random.turn` and `random.move`, with domain sizes of 3 and 2, and robot variables `last_num_con`, with domain size 3. By representing D_v as the domain size of variable v we derive the total number of states for each encoding as being $(D_x \times D_y \times D_{direction} \times D_{motion})^3 \times D_{turn}$ for the global abstraction in [7, 8], $(D_x \times D_y \times D_{direction} \times D_{motion})^2 \times (D_{motion}) \times D_{turn} \times D_{random}$ for our relative encoding of the same abstraction, and $(D_x \times D_y \times D_{direction})^2 \times D_{turn} \times D_{rand.turn} \times D_{rand.move} \times (D_{last_num_con})^3$ for our new abstraction. The result of those calculations are mirrored by the values shown in Table 2.

Table 2: State space size (approximate) for different encodings of the *Alpha* algorithm

Concurrency	Statistic	Abstraction from [7, 8]		Our new abstraction Relative
		Global	Relative	
Fair asynchrony	Total States	134.2×10^6	1.1×10^6	31.9×10^6
	Reachable States	68.1×10^6	0.5×10^6	1.6×10^6
Strict turn taking	Total States	402.7×10^6	3.1×10^6	31.9×10^6
	Reachable States	1.1×10^6	0.2×10^6	0.4×10^6
Non-strict turn taking	Total States	2818.5×10^6	22.0×10^6	223.0×10^6
	Reachable States	48.4×10^6	0.7×10^6	1.3×10^6

Figure 1 and Figure 2 show a comparison of the state-space for global and relative encodings from the abstraction implemented in [7, 8], and our new abstraction when varying the grid size or the number of robots, when compiling the

models in NuSMV. These experiments consider a strict turn taking concurrency mode. We repeated these experiments for other concurrency modes, non-strict turn taking and fair asynchrony, with similar results in the state-space reduction. In the experiments of Figure 1, the number of robots is set to 3, and the size of the grid is varied. In the experiments of Figure 2, the size of the grid is set to 6×6 , and the number of robots is varied. The final points in the graph correspond to the limit in the memory for the verification to be possible.

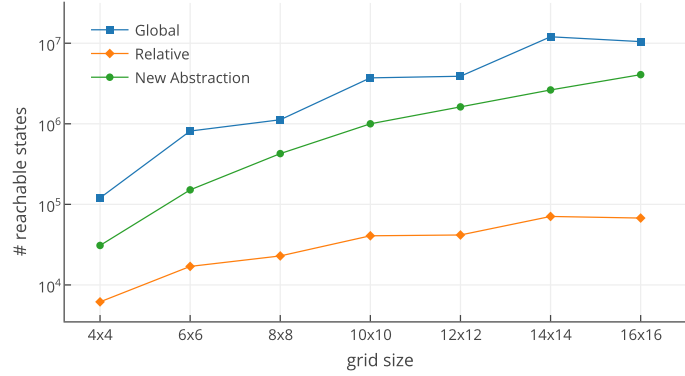


Figure 1: Reachable states as the grid size increases, for strict turn taking concurrency mode and 3 robots

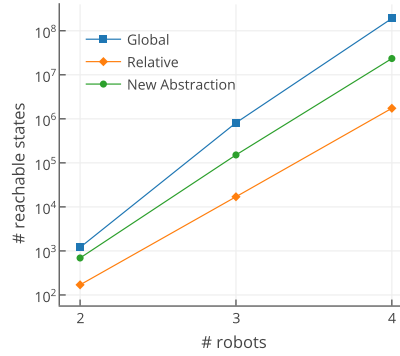


Figure 2: Reachable states as number of robots increases, for strict turn taking concurrency mode and a 6×6 grid

We verified the LTL property mentioned previously, *all the robots will eventually be connected*, in all these models, to validate the relative encoding of the abstraction in [7, 8], with respect to their original global model. The verification results were identical, indicating our relative encoding preserved the properties of the global encoding. For the verification, we used NuSMV version 2.5.4, running on a PC with Ubuntu 14.10 and 4 GB RAM.

A counterexample (failing trace) provided by the model checker, with strict turn taking concurrency mode, 3 robots, and a grid of size 5×5 , is shown in Figure 3, for the global encoding. From state 5 onwards, all robots move south in a loop and robot C (circle) never reconnects to the swarm. In the global model, it takes 15 states until the loops starts. However, the same pattern is observed within each 3 steps from the moment when robot C (circle) changes its direction. This repetitiveness was eliminated in the relative model, as illustrated by Figure 4, achieving a reduction of 12 states. In the relative encoding, the location of the reference (circle) is fixed to cell $(0,0)$ and its direction to North. Subsequently, other robots update their position and orientation according to the decisions of the reference robot, or according to their individual decisions, as explained in Section 4.

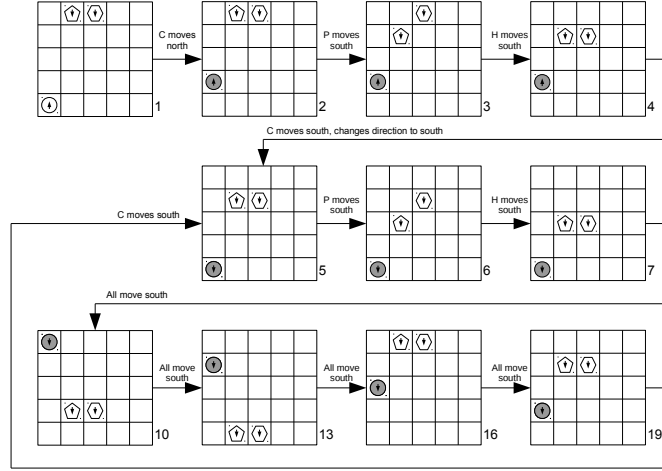


Figure 3: Failing trace of globally encoded model. C: circle, P: pentagon, H: hexagon. Disconnected robots in gray

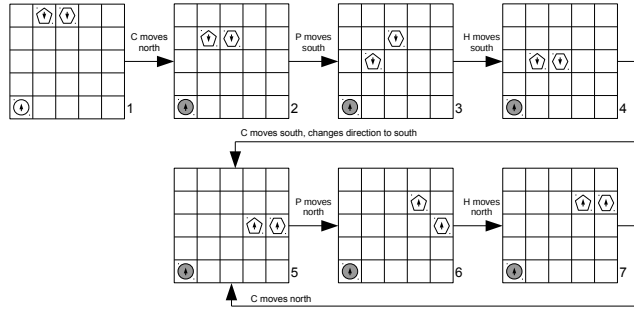


Figure 4: Failing trace of relatively encoded model. C: circle (reference), P: pentagon, H: hexagon. Reference: circle. Disconnected robots in gray

Posteriorly, we compared the verification results using different abstractions of the *Alpha* algorithm, the relative version of the one proposed in [7, 8] and our new abstraction, also relative. These experiments were conducted for $\alpha = 1$, and a strict turn taking concurrency mode. We found that the LTL property is true for some settings, such as 3 robots in grids of 2×2 , 3×3 and 4×4 , and two robots in a 5×5 grid. For other settings, such as 3 robots in a 5×5 grid, the property is false and the swarm will not regain connection. This is caused by the randomness of the individual robots decisions, that can be repeated infinitely, i.e., a robot can infinitely often perform the same patterns of motion but they do not lead to regain connectivity.

Figure 5 and Figure 6 show the verification time for the LTL property in Section 5 for global and relative encodings from the abstraction [7, 8], and the new abstraction when varying the grid size or the number of robots. These experiments, as before, consider a strict turn taking concurrency mode. In the experiments of Figure 5, the number of robots is set to 3, and the size of the grid is varied. In the experiments of Figure 6, the size of the grid is set to 6×6 , and the number of robots is varied. The final points in the graph correspond to a stipulated time limit of 5 days for the verification.

We observed the same time reduction patterns between the global and relative encodings of the abstraction in [7, 8], as a consequence of the state-space reduction. The number of robots is a more significant constraint to the verification time than the grid size. Although it was not possible to verify the relative encoding with 4 robots due to time restrictions, applying some constraints to the initial configuration of the swarm allowed the generation of a counterexample, a proof that the property is false for that number of robots and grid size. In contrast, the global encoding could not be verified, even when applying the same constraints. The relative encoding of the new abstraction of the *Alpha* algorithm takes longer to be verified than the previous abstraction, as it is more complex. Nevertheless, it allowed us to obtain different verification results compared to [7, 8], which we believe are closer to the intention of the *Alpha* algorithm. The difference between the two abstractions of the *Alpha* algorithm need to be further investigated to determine if any are incorrect, given the verification results.

7 Conclusions

We presented an approach that achieves significant state-space reduction and thus allows model checking more complex emergent behaviours of swarms. We propose the use of symmetry reduction to eliminate symmetrical states (i.e., configurations of robots in the grid), implemented through a relative encoding of the swarm, where one robot is the reference, and the others' navigation is encoded relative to it. This encoding, compared to a global one, helps to reduce the state space of the model, as demonstrated by our results in Section 6. Thus, verification through model checking can be performed over larger grid sizes and higher numbers of robots, and also for more detailed abstractions that model

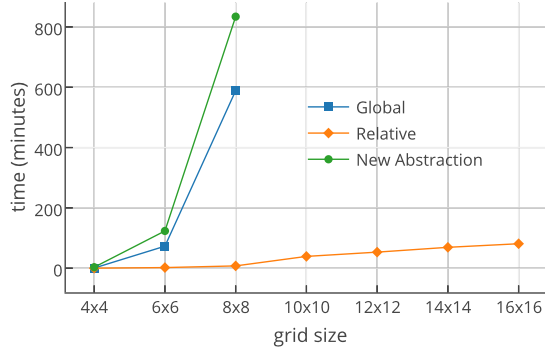


Figure 5: Verification time for the LTL property in Section 5 as the grid size increases, for strict turn taking concurrency mode and 3 robots

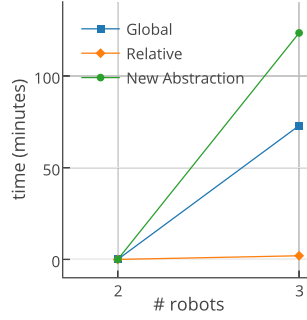


Figure 6: Verification time for the LTL property in Section 5 as number of robots increases, for strict turn taking concurrency mode and a 6×6 grid

navigation algorithms using more variables. Although the state space reduction is more significant in terms of the grid size, and not as expressive if considering realistic swarm sizes, analysing small robot groups can help to understand larger swarms within the lower limit bounds of swarm size, which demands more from the navigation algorithm [7, 8].

Future work includes running more experiments with different α values, and the verification of more complex properties of robotic swarms, such as the emergence of teams or different swarms due to the connectivity properties. The successful relative encoding of a new abstraction of the *Alpha* algorithm gives us confidence that our approach can be applied to other swarm algorithms. In the future we would like to model navigation algorithms such as the *Beta* algorithm [17] in a relative manner, to validate the transferability of our approach.

Acknowledgements We would like to thank Clare Dixon for providing her *Alpha* algorithm models, and her invaluable comments and advice. The work by D. Araiza-Illan and K. Eder was partially supported by the EPSRC, grants

References

- [1] Appold, C.: Efficient symmetry reduction and the use of state symmetries for symbolic model checking. In: Proc. GandALF. pp. 173–187 (2010)
- [2] Bosnacki, D., Dams, D., Holenderski, L.: Symmetric SPIN. In: SPIN Model Checking and Software Verification. pp. 1–19. Stanford, CA, USA (2000)
- [3] Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV version 2: an open-source tool for symbolic model checking. In: Computer-Aided Verification. pp. 359–364. Copenhagen, Denmark (2002)
- [4] Clarke, Jr., E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge, MA, USA (1999)
- [5] Clarke, E., Enders, R., Filkorn, T., Jha, S.: Exploiting symmetry in temporal logic model checking. Formal Methods in System Design 9(1–2), 77–104 (1996)
- [6] Dixon, C., Winfield, A.F., Fisher, M.: Verification of swarm robots: the Alpha algorithm. In: Proc. ARW. pp. 10–11. Westminster, UK (2010)
- [7] Dixon, C., Winfield, A.F., Fisher, M.: Towards temporal verification of emergent behaviours in swarm robotic systems. In: Towards Autonomous Robotic Systems. pp. 336–347. Sheffield, UK (2011)
- [8] Dixon, C., Winfield, A.F., Fisher, M., Zeng, C.: Towards temporal verification of swarm robotic systems. In: Robotics and Autonomous Systems. pp. 1429–1441. Bristol, UK (2012)
- [9] Donaldson, A.F., Miller, A.: Automatic symmetry detection for model checking using computational group theory. In: Formal Methods. pp. 481–496. Newcastle, UK (2005)
- [10] Emerson, E., Wahl, T.: On combining symmetry reduction and symbolic representation for efficient model checking. In: Correct Hardware Design and Verification Methods. pp. 216–230. LAquila, Italy (2003)
- [11] Emerson, E., Wahl, T.: Dynamic symmetry reduction. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 382–396. Edinburgh, UK (2005)
- [12] Gomes, P.d.C.: Verification of symmetric models using semiautomatic abstractions. Master’s thesis, Computer Science, Belo Horizonte, Brazil (2010)

- [13] Juurik, S., Vain, J.: Model checking of emergent behaviour properties of robot swarms. *Proceedings of the Estonian Academy of Sciences* 60(1), 48–54 (2011)
- [14] Kloetzer, M., Belta, C.: Temporal logic planning and control of robotic swarms by hierarchical abstractions. *IEEE Transactions on Robotics* 23(2), 320–330 (2007)
- [15] Konur, S., Dixon, C., Fisher, M.: Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems* 60, 199–213 (2012)
- [16] Kouvaros, P., Lomuscio, A.: A counter abstraction technique for the verification of robot swarms. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, January 25–30, 2015, Austin, Texas, USA. pp. 2081–2088 (2015)
- [17] Nembrini, J.: *Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots*. Ph.D. thesis, Bristol, UK (2005)
- [18] Norris IP, C., Dill, D.L.: Better verification through symmetry. *Formal Methods in System Design* 9(1-2), 41–75 (1996)
- [19] Rouff, C., Hinchey, M., Truszkowski, W., Rash, J.: Formal methods for autonomic and swarm-based systems. In: *Proc. ISoLA*. pp. 100–102. Paphos, Cyprus (2004)
- [20] Winfield, A.F., Sa, J., Fernandez-Gago, M.C., Dixon, C., Fisher, M.: On formal specification of emergent behaviours of swarm robotic systems. *International Journal of Advanced Robotic Systems* 2(4), 363–370 (2005)